International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2026/12 Volume 13, Issue 1, January – February (2026)

RESEARCH ARTICLE

Carbon and Energy-Aware Cloud Resource Management and Task Scheduling Framework

Mohammed A. S. Mosleh
Department of Computer Science and Engineering, Al-Fayha Private College, Jubail Industrial City, Saudi Arabia.
✉ m.mosleh@fayha.edu.sa

Received: 20 October 2025 / Revised: 30 January 2026 / Accepted: 12 February 2026 / Published: 26 February 2026

Abstract – The exponential growth of cloud computing has intensified energy consumption and carbon emissions, posing a critical challenge to environmental sustainability. To address this, the present study proposes a novel CarbonLiteFormer-GSBO framework for energy- and carbon-aware cloud resource management and task scheduling. The proposed CarbonLiteFormer, a lightweight adaptive Transformer architecture, integrates an Energy–Carbon Dual Attention (ECDA) mechanism that dynamically learns correlations between resource utilization, carbon footprint, and workload complexity. Unlike conventional deep or static schedulers, CarbonLiteFormer leverages hybrid spatial-temporal embeddings to predict task demands while maintaining minimal model complexity. The optimization phase employs the Genetic Secretary Bird Optimization (GSBO) algorithm, a synergistic metaheuristic that combines the global search capability of Genetic Algorithms with the adaptive exploration of the Secretary Bird Optimizer to achieve cost-effective and energy-efficient scheduling decisions. The framework was evaluated using two benchmark datasets, the CloudSim plus Task Scheduling Dataset and the PlanetLab Energy-Aware Workload Trace Dataset, under multiple QoS parameters such as makespan, throughput, energy efficiency, and carbon emission index. Experimental results demonstrate that CarbonLiteFormer-GSBO outperforms existing state-of-the-art models, achieving 23.8% lower energy consumption, 18.6% carbon reduction, and 16.4% improvement in scheduling efficiency. The proposed approach provides a sustainable, scalable, and adaptive solution for next-generation green cloud computing environments.

Index Terms – Energy-Aware Scheduling, Carbon Emission Optimization, Lightweight Transformer, CarbonLiteFormer, Genetic Secretary Bird Optimization, Sustainable Cloud Management.

1. INTRODUCTION

The digital infrastructure has been transformed due to increase in cloud computing, which provides scalable and on-demand computational services [1]. However, the cloud service demand has increased data centre energy consumption (EC) and carbon emissions. The current data centres approximately increase global electricity consumption and carbon footprint by 1-2% due to IOT workloads and big data systems. This growth increases operational cost and contributes to greenhouse gas emissions. To address this limitation, an efficient and adaptive scheduling algorithm is developed with carbon and energy constraints to balance Quality of Service (QoS) [2]. Current optimization models are limited in addressing balance between joint energy, carbon and QoS. Conventional Metaheuristic and heuristic algorithms, including Particle Swarm Optimization (PSO) and hybrid models, are better for static task allocation but are limited in adapting to dynamic workloads [3]. The DL schedulers are computationally heavy, lack poor generalization with unknown patterns and do not consider carbon emission constraints modelling. Transformer scheduling modules learn temporal patterns, but have high computational complexity and limited interpretability. The task scheduling frameworks lack joint optimization models that decrease EC and carbon emission that preserves QoS requirements [4]. The models are limited in explainable and adaptive AI models, capable of interpreting the trade-offs between resource utilization, power efficiency and emission metrics.

Modern hyperscale data centres host millions of virtual machines and execute billions of tasks daily, resulting in substantial power demand and increased operational costs. This escalating EC directly contributes to greenhouse gas emissions in regions in which electricity generation is dominated by fossil fuels. Carbon-aware cloud scheduling introduces additional complexity because carbon intensity varies spatially and temporally depending on power grid conditions, renewable energy penetration and regional energy policies.

DL–based schedulers incur high training and inference costs and metaheuristic algorithms struggle to scale efficiently under dynamic workloads and heterogeneous cloud infrastructures. The lack of lightweight predictive models capable of capturing energy–carbon–workload correlations limits the practicality of deploying such solutions in operational cloud platforms where low latency and stability are critical. The current models have high computational overhead, which restricts practical deployment in real-time and resource-constrained cloud environments [5]. These gaps

RESEARCH ARTICLE

highlight the necessity for a hybrid, interpretable and computationally lightweight framework that adapts dynamically to workload variations and addresses environmental sustainability goals.

This research developed an AI–metaheuristic hybrid framework called CarbonLiteFormer-GSBO that integrates a lightweight Transformer model with a Genetic Secretary Bird Optimization (GSBO) algorithm for sustainable cloud task scheduling. The CarbonLiteFormer-GSBO model includes Energy–Carbon Dual Attention (ECDA) to gather complex correlations between workload, energy utilization and carbon footprint. This model enables the model to classify energy-efficacy and lower-carbon task allocation. The GSBO integrates GA mechanisms for exploration and SBO adaptive behaviour for exploitation for tuning the scheduling solutions. The CarbonLiteFormer-GSBO combines the lightweight Transformers with the adaptive search capabilities for better scheduling. The objective of this research is to evaluate a sustainable, efficient and adaptive cloud task scheduling framework that jointly minimizes EC and carbon emissions while maintaining high QoS. This study aims to:

- To develop a lightweight transformer-based prediction model that accurately captures workload dynamics, resource utilization patterns, and time-varying carbon intensity with low computational overhead.
- To design an ECDA to explicitly model the interdependence between EC and carbon footprint during task execution.
- To integrate a hybrid metaheuristic GSBO algorithm that balances global exploration and local exploitation to achieve stable and fast convergence under dynamic workloads.

The major contributions of this work are summarized as follows:

- Carbon- and Energy-Aware Lightweight Transformer (CarbonLiteFormer) incorporates ECDA to dynamically gather energy-carbon-QoS correlations, which enables low complexity in cloud environments.
- A hybrid metaheuristic that combines Genetic Algorithm crossover–mutation exploration with the Secretary Bird's adaptive pursuit strategy to enhance global convergence and adaptive decision-making.
- A comprehensive scheduling framework balances QoS delivery, cost-efficiency and eco-friendly task execution, demonstrating reductions in energy consumption and carbon emissions.

The remainder of this paper is structured as follows: Section 2 reviews the related work in energy and carbon-aware task scheduling. Section 3 presents the architecture and functional modules of the CarbonLiteFormer-GSBO framework. Section 4 details the experimental setup, datasets, evaluation metrics and comparative analysis with existing models and Section 5 concludes with key findings and future research directions.

2. RELATED WORK

2.1. Energy and Carbon-Aware Scheduling

Recent advances have leveraged time-series forecasting, renewable energy prediction and intelligent VM allocation to align computing workloads with green energy availability. Techniques such as ensemble decomposition, deep learning-based carbon prediction, and green core resource modeling enable more proactive and sustainable decision-making in cloud environments.

Zhao et al. [6] proposed an Energy and carbon-aware algorithm (ECA) with Predictive renewable energy (EFP) model. The model gathered EC features, renewable energy features, carbon footprint features and temporal features. The EFP includes the Ensemble Empirical Mode Decomposition (EEMD) to decompose non-linear wind speed time-series into multiple intrinsic mode functions (IMF) components. The temporal convolutional network (TCN) predicts each decomposed component independently. The ARIMA model predicts CPU utilization and workload patterns and aggregates the prediction. The model is simulated using Google Cluster Data (GCD) and attained 0.545 RMSE, 73.11% utilization, 322869.5W energy consumption and 0.2% SLA violation. The model balances energy efficiency and carbon reduction objectives. Yet, the model has higher VM migration and data transfer latency costs. Beena et al. [7] presented an ECA Algorithm integrated with LSTM-based Carbon Intensity Prediction Model. The collected data is cleaned, normalized and missing values are handled and the Z-score is used for outlier detection. The model used LSTM for time-series forecasting, ARIMA for statistical modeling and exponential smoothing for seasonality adjustments. The model is evaluated using carbon intensity data from electricity maps platform. The model attained 427.5 kWh average carbon intensity. The modular architecture is easy to maintain and extend with new features. Yet, the model fails on service quality and has impaired decision-making capability. Hewage et al. [8] proposed an OpenStack-GC Framework that integrated VM execution model and VM packing algorithm, renewable-driven cores (RDC) and green cores (GC) concept. The server-level renewable energy is allocated through mixed power delivery system and homogeneous server configuration with grid and renewable energy capacities. The GC converts renewable energy usage into server inventory attributes and server-level VM execution model dynamically switches between real-time and low-power profiles and at data center-level VM packing algorithm and servers are plotted in 2D Euclidean space using feature vectors. The RDC dynamically adjusts per-core power profiles based on renewable energy

RESEARCH ARTICLE

availability. The model is simulated on OpenStack cloud infrastructure and attained 6.52 latency and 59W power. GC enables lightweight VM packing decisions suitable for data center scale. Yet, the model degrades in dynamic VM workloads.

Although ECA scheduling strategies demonstrate substantial progress toward sustainable cloud computing, they face several practical and methodological limitations, including increased migration and latency costs, fail to maintain service quality, leading to suboptimal decision-making during scheduling and have limited adaptability to dynamic workloads.

2.2. Metaheuristic Optimization in Cloud Scheduling

Various metaheuristic frameworks have been developed by using evolutionary processes, swarm intelligence, physical phenomena and hybridized strategies. These methods focus on improving aspects of global search, local refinement, diversity maintenance and multi-objective trade-offs, through hybridization or adaptive mechanisms. These approaches have shown improvements over classical scheduling methods in efficiency and solution quality.

Tanha et al. [9] developed a Genetic Algorithm and Thermodynamic Simulated Annealing (GATSA) model. The GATSA combined GA global exploration capability with TSA's local exploitation strength. The model used semi-conducted population initialization and Temperature adjustment based on solution quality using thermodynamic laws and entropy concepts. The Annealing process accepts solutions probabilistically based on Boltzmann factor and the temperature is updated using entropy and energy difference. The model is simulated using Molecular dynamics task graph with 41 nodes and attained 126s makespan, 440.34s execution time, 5.75% SLR and 15.534 efficiency value. The model used thermodynamic laws for intelligent temperature reduction that increased adaptive cooling schedule. Yet, the model has higher complexity than list schedulers. Alsadie et al. [10] proposed a Metaheuristic Framework for Dynamic Virtual Machine Allocation (MDVMA). The MDVMA used NSGA-II for true multi-objective optimization and provided Pareto front of non-dominated solutions. The task allocation module assigns selected non-inferior schedule to VM and the diversity is preserved using crowding distance function. The model is simulated on CloudSim and attained 31.93Kwh energy usage, 7338s makespan and 7794 cost. The Pareto Front solutions balance multiple non-inferior solutions. Yet, the model has initialization and population management overhead. Pirozmand et al. [11] presented a Multi-Adaptive Learning for Particle Swarm Optimization (MALPSO) model. The MALPSO enhances PSO through adaptive population division that divides population through clustering and chaos-based inertia weight that enables variable search steps. The PSO is modified with adaptive learning mechanisms and the task-to-resource assignment is optimized. The model is evaluated using CEC 2017 and attained 4987s makespan, 3261s time, load balancing index of 14 and efficiency of 0.15. The model provided better solution quality. Yet, the model increases computational overhead due to clustering step. Iyappan et al. [12] proposed a Hybrid Simulated Annealing and Spotted Hyena Optimization Algorithm (HAS-SHOA) model. Tasks are scheduled to VM using SHOA to maintain balanced load distribution. SHOA explores new resource allocation configurations and SA refines worse solutions with probability. Cloud resources are allocated and managed using a combination of SA and SHOA for dynamic resource management. The model is simulated on PySim and attained 350.21ms response time, 400kW EC and 6498s execution time. The model increased load-balancing stability. Yet, the hybrid algorithm involves multiple stages and surplus resource tracking. Mansour et al. [13] presented an Energy Efficient Resource Scheduling using Cultural Emperor Penguin Optimizer (EERS-CEPO). The CA provides knowledge-based guidance through situational and normative knowledge and EPO simulates emperor penguin behavior with Huddle boundary generation and temperature profile calculation. The CA enhances EPO's exploitation capabilities by guiding population development using stored knowledge. The model is simulated on CloudSim and attained 1860ms response time, 5403 execution time, and 2.45kWh energy consumption. The model used knowledge reuse for solution diversity. Yet, the hybrid algorithm introduces additional computational overhead.

Mostafa et al. [14] presented a Modified White Shark Optimizer (mWSO) for Cloud Task Scheduling. The mWSO includes memory-based method to focus on global best solution and enhanced velocity update equation focusing on global best position. The model used new position updating strategy with three movement conditions based on random parameters and used exponential time transfer mechanism. The model is evaluated on CloudSim and attained 135.3s makespan and 32.222W energy consumption. The model is effective for static scheduling. Yet, the model doesn't handle dynamic task arrivals during execution.

Pabitha et al. [15] proposed a Chameleon and Remora Search Optimization Algorithm (CRSOA) model. The CRSOA combined Chameleon Search Algorithm (CSA) and Remora Search Optimization Algorithm (RSOA) using greedy methodology for managing uncertainty. The model is simulated using CloudSim and attained 125s makespan, 42546s cost and 0.15% degree of imbalance. The model is efficient in uncertain task distributions. Yet, the model is limited in handling sudden workload changes in real-time. Tiwari et al. [16] presented a Neighborhood-Inspired Multiverse Scheduler (NIMS). The NIMS used fitness-based neighborhood search strategy during solution updates, Roulette wheel selection for VM selection to maintain

RESEARCH ARTICLE

diversity and neighborhood selection probability (NSP) that decreases linearly over iterations. The model is simulated using CloudSim and attained 191.4s makespan, 2.12kWh energy consumption and 7.3mbps throughput. The model balanced randomness and guided search. Yet, time complexity increases with neighborhood size and number of VM. Rostami et al. [17] presented a Hybrid Multi-Objective Capuchin Search Algorithm and Inverted Ant Colony Optimization (CapSA-IACO) approach. The task is assigned based on IACO algorithm with pheromone-based probabilistic selection and the VM is migrated using CapSA process. The model is simulated on MATLAB and attained 81.23% load balancing, 2.547kwh power consumption, 0.74min execution time and 1842s makespan. The model effectively separates scheduling and migration tasks. Yet the model requires careful parameter tuning. Tabagchi Milan et al. [18] proposed a QoS-based Load Balancing by using ABC Algorithm (QLBABCA) model. The QLBABCA calculates VM capacity using processor numbers and MIPS and determines EC using dynamic power dissipation. The model decides load balancing using standard deviation and uses the HeapSort algorithm for task sorting based on QoS.

The model is simulated on CloudSim and attained load of 6531.2 and power consumption of 810J. The model is effective for energy modeling. Yet, the model lacks in highly dynamic environments with rapid workload changes. Kumar et al. [19] presented a Greylag Sand Cat Swarm Optimization Algorithm (GSCOA) model. The GGOA imitates V-formation and leadership dynamics of geese during migration to enable broad search space exploration and uses SCSOA to focus on refining solutions. The model is evaluated on HPC2N and attained 31.34 kWh energy consumption, 4335.6 cost and 12.536 carbon emission factor. The model is suitable for carbon-aware optimization. Yet, this process increases scheduling time for large-scale. Barut et al. [20] presented a Parallel Hybrid Multi-Objective Metaheuristic Task Scheduling Approach combining NSGA-2 and SPEA2 algorithms. The NSGA-2 Algorithm used non-dominated sorting for candidate grouping and simulated binary crossover (SBX) for genetic operations. The SPEA2 Algorithm used external archive for elite solutions and k-nearest neighbor density estimation. The Parallel Hybrid Execution runs multiple algorithm instances runs on separate processor cores. The model is simulated on CloudSim and attained 642.25s makespan and 689,254.71W energy. The model preserves solution diversity through parallel execution. Yet, the parallel architecture demands high-capacity computational infrastructure. Malik et al. [21] proposed a Lateral Hyena-Particle Swarm Optimization (LH-PSO) algorithm. The PSO parameters and particles are initialized with position and velocity values and LH operation updates fitness values through population initialization, applying a lateral method to update each searching agent's fitness value and recording the best value. The model is simulated on CloudSim and attained 12.97s response time and 91% memory utilization. The model has faster convergence. Yet, the model lacks performance in dynamic environment. Khaledian et al. [22] proposed a Two-Stage Task Scheduling Approach Using TOPSIS and Multi-Objective Archimedes Optimization Algorithm (TM-MOAOA). The tasks are prioritized using Multi-Criteria Decision-Making (MCDM) and the prioritized tasks are scheduled using MOAOA. The TOPSIS calculates proximity using geometric distance. The model is simulated on CloudSim and attained 32.18s makespan, 10s execution time, 7.42% speedup improvement and 301.9Wh energy consumption. The model handles multiple objectives with clear decision hierarchy. Yet, the model lacks fault tolerance gaps. Mikram et al. [23] presented a Chi-squared PSO (CHPSO) model. The CHPSO algorithm used chi-squared distribution for task arrival modeling, heuristic load balancing strategy balances workload and PSO-based task assignment optimization to refine task-to-VM mapping, ensuring optimal task placement. The model is simulated on CloudSim and attained 6550s makespan, 74.8% resource utilization and 1.35M energy consumption. The model combines deterministic and stochastic optimization for better performance. Yet, the heuristic load-balancing increased computational overhead.

Nandagopal et al. [24] proposed a Hybrid Cuckoo Search and with Modified BERT-Based Transformer for Energy-Efficient Cloud Computing. In the preprocessing step, the missing values are handled and normalized using min-max scaling. The HCS is used for dynamic task priority allocation and the BERT is modified with self-attention for classification. The model is validated using the Kaggle dataset and attained 0.430J EC, 375ms execution time and 485ms Makespan. The model has better resource utilization. Yet, the model has communication overhead and causes latency issues. Ahmed et al. [25] proposed an Intelligent Learning-based Cloud Task Scheduling (ILbCTS) algorithm. The collected data is processed for handling missing values. The deep reinforcement learning model is used with dynamic adaptation for changing states and available rewards. The model is validated using Job sets with 1000, 5000 and 10,000 tasks and attained 402ms execution time. The model has adaptive learning capability. Yet, the model lacks in handling dynamic workloads. Choudhary et al. [26] proposed an enhanced modified Min-Min using Cuckoo Search Algorithm. The data handled missing values and normalized using scaling. The Min-Min phase calculates the minimum completion time and assigns the tasks to VM. The CS with Levy flight is used for optimizing values. The model is validated using Cybershake workflow and attained 512ms makespan and 76% resource utilization. The model combined strengths of Min-Min and Cuckoo Search. Yet, the model performance lacks in dynamic cloud environments.

RESEARCH ARTICLE

Rajeshwari et al. [27] proposed a Linked List-based Push and Pull (LPP) Task Scheduling. The preprocessing step identifies task priority based on user SLA information stored in the database and calculates deadline for each incoming task. In the task assignment process, each VM analyses the linked list node and pulls the task that can be completed within deadline based on current resource capacity. The task is assigned based on deadline feasibility per VM. The model is validated using different test loads on CloudSim and attained 98.72% SLA violation rate, 3490ms waiting time and 3506ms completion time. The model completed tasks within deadline constraints. Yet, the model lacks in handling heterogeneity VM. Kumar et al. 28] proposed a Deep Neural Network with Improved Grey Wolf–Horse Herd Optimization (DNN-IGW-HHO) model. The tasks are initialized and are characterized by feature vectors. The DNN model includes input, activation and output layers are used for classification. The IGW-HHO algorithm is used for tuning the parameters. The model is experimented on different task loads and attained 1407.1ms execution time. The model minimizes task completion time. Yet, the model lacks performance in dynamic and heterogeneous environments.

Metaheuristic optimization modules have challenges, such as high computational complexity and initialization overhead. This process increases processing cost and limited adaptability to dynamic workloads, which fail to manage real-time task arrival variations and demand high-capacity computing resources.

2.3. AI-Based Cloud Resource Management

Recent research has leveraged a wide range of AI paradigms, such as graph neural networks, ANN and transformer architectures to tackle various aspects of cloud task scheduling, energy optimization, and QoS assurance. These approaches learn from historical patterns and optimize multi-objective trade-offs effectively than conventional algorithms.

Tuli et al. [29] presented a Holistic Resource Management technique for sustainable cloud computing (HUNTER) model. The HUNTER used Gated Graph Convolution Network with GRU-based weighting to model inter-task dependencies. A top-k heuristic scheduling module selects the most suitable VMs based on predicted task influence and system state. This graph-based learning enables accurate modeling of complex task interactions. The model is simulated on COSCO Framework and attained 908s training time, 221KWh energy and 1.1% SLA violations. The approach improves SLA compliance and reduces EC through informed dependency-aware scheduling. Yet, the model does not handle task demands. Zavieh et al. [30] proposed an Artificial Neural Network Dynamic Balancing (ANNDB) model. The ANNDB model leveraged feed-forward network architecture and multi-layer perceptron (MLP) for optimal task allocation. The neural network predicts balanced resource distribution by modeling nonlinear relationships between workload features and VM capacities. Dynamic balancing decisions are generated in real time during scheduling. The model is simulated using CloudSim and attained 99% accuracy and improved 13.81% energy and 4.84% power criterion. The model achieves high prediction accuracy and notable improvements in energy and power efficiency. Yet, the model has increased computational complexity due to an increased number of layers. Mangalampalli et al. [31] introduced a DRL-based TS algorithm (DRLBTSA) to efficiently schedule tasks onto VM in cloud computing environments. The DRLBTSA algorithm used a Deep Q-network (DQN) based on RL to make adaptive scheduling decisions based on task priorities and resource availability. DRLBTSA formulates cloud scheduling as a Markov Decision Process and applies a Deep Q-Network to select optimal VM assignments based on system state and task priority. The agent learns scheduling policies through continuous interaction with the cloud environment using reward feedback. The model simulations are performed using CloudSim and real-world HPC workloads. The model attained makespan by up to 45.5%, energy consumption by up to 53.2% and SLA violations by up to 61.1%. The model reduces makespan, energy consumption, and SLA violations. Yet, the model increased the computational overhead due to Deep Q-Network. Slathia et al. [32] proposed a SHAP-Enhanced Resource Allocation for VM Scheduling (SHERA) model. The SHERA model used random forest classifier for feature importance by constructing multiple decision trees. The features are aggregated for final prediction and optimizing VM scheduling. The SHAP is integrated to quantify feature importance and provide explainable scheduling decisions. The ensemble learning structure enhances prediction stability and robustness. The model is simulated on Cloud Efficiency Dataset from Google Cloud and attained 0.0256 MSE, 0.16 RMSE and 96.8% accuracy. This model improves allocation accuracy and interpretability in resource management. Yet, the model has increased processing time. Arulkumar et al. [33] presented a Privacy-Oriented White Shark Encompassed hierarchical auto-associative polynomial CNN (POWSE-HAP-CNN) model. The HAP-CNN model used polynomial convolution with auto-associative learning to gather complex non-linear characteristics and the parameters are optimized using WSO. The privacy mechanisms include direct and indirect trust with risk probability based on security demands. Trust evaluation and risk probability mechanisms enforce privacy-aware task execution. The model used Google Cloud Jobs (GoCJ) dataset and attained 99.32% accuracy, 97.12% precision, 95.93% recall, 96.15% F1 score, 48.3s makespan, 5.81s response time, 20.76 risk probability and 96.56% trust values. This architecture achieves high scheduling accuracy while ensuring secure and trustworthy cloud resource utilization. Yet, the polynomial expansions in HAP-CNN increase computational cost and parameter count. Nandagopal

RESEARCH ARTICLE

et al. [34] proposed a Hybrid Cuckoo Search with Modified BERT-Based Transformer (HCS-BERT) model. The HCS-BERT model used HCS for dynamic task priority allocation and modified BERT-based transformer used attention mechanism for contextual learning and learns from historical scheduling patterns. This hybrid framework learns temporal workload patterns to guide scheduling policies.

The model is stimulated using cloud computing performance metrics dataset and 0.430J energy consumption, 380ms execution time, 0.0044J/task energy efficiency, 88.2% resource utilization and 490ms makespan. It provides fast execution, high resource utilization, and strong energy efficiency. Yet, the model has communication overhead and increases latency. Sharma et al. [35] presented a Quantum Tensor-based Deep Neural Network and Binary bird swarm optimization (QT-DNN-BBSO) algorithm. The QT-DNN-BBSO model used the features from VM. The features are processed using three-layer architecture and the softmax layer produces probability distribution. The QT integrates Poisson process for task arrivals and BBSO is used for dynamic workload redistribution between overloaded and underloaded VM. The model is evaluated using synthetic dataset and attained 98.12% accuracy, 97.35% precision, 97.27% recall, 88.1% resource utilization and 97.49% F1 score. The approach enhances prediction accuracy and resource utilization under dynamic scheduling conditions. Yet, the model has increased computational resources and doesn't handle heterogeneous workloads.

AI-based resource management methods exhibit potential for adaptive, sustainable and efficient cloud scheduling, but the models lack in trade-offs between accuracy, interpretability, scalability and computational cost. These limitations motivate the need for lightweight yet powerful hybrid frameworks to balance predictive intelligence with optimization efficiency for real-time sustainable cloud scheduling.

2.4. Identified Limitations

The literature review highlights developments in ECA scheduling, metaheuristic optimization and AI-driven resource management for sustainable cloud computing. Although these approaches have demonstrated improvements in performance metrics such as makespan, energy efficiency, SLA compliance and carbon reduction, these models have research gaps that limit their effectiveness in large-scale, real-time and dynamic cloud environments.

- Many existing metaheuristic and AI-based scheduling frameworks, such as GATSA, MALPSO, HCS-BERT and QT-DNN-BBSO, involve multi-stage hybridization, deep neural architectures and parallel algorithm execution. These models increase time complexity, memory consumption and parameter tuning, which makes real-time deployment challenging in large cloud data centers.
- Several methods such as mWSO, CRSOA and OpenStack-GC, demonstrate limited responsiveness to sudden workload changes, dynamic task arrivals and heterogeneous resource configurations.
- The ECA scheduling methods in AI-driven scheduling frameworks, such as ANNDB, DRLBTSA, SHERA and HCS-BERT are limited in carbon footprint constraints that create gap between task allocation and sustainability objectives.

2.5. Problem Statement

The rapid growth of cloud computing has led to an increase in data center energy consumption and carbon emissions, raising operational costs and environmental impact. Existing task scheduling frameworks are limited in simultaneously addressing energy efficiency, carbon footprint and QoS. Metaheuristic algorithms fail to adapt to dynamic workloads, while DL–based schedulers, including Transformer models, struggle with poor generalization to unseen workloads, high computational overhead and limited interpretability. Most frameworks do not explicitly account for the temporal variability of carbon intensity in power grids or integrate predictive intelligence with optimization strategies. These limitations hinder the deployment of real-time, sustainable and scalable cloud scheduling solutions. There is a need for a lightweight, adaptive and carbon-aware scheduling framework.

2.6. Research Gaps

The research gaps include

- Most existing scheduling frameworks optimize energy consumption without explicitly modeling the temporal variability of carbon intensity in power grids.
- DL–based schedulers such as DRL, BERT-based Transformers, and hybrid CNN architectures achieve high accuracy but introduce substantial training cost, inference latency, and memory overhead.
- Existing frameworks separate workload prediction and scheduling optimization and lack of integration between predictive intelligence and optimization strategies leads to slower convergence, inefficient search behavior, and unstable scheduling decisions under rapidly changing system states.

The review indicates that existing energy and carbon-aware scheduling frameworks struggle with several critical limitations that restrict their applicability in real-world cloud environments. These limitations highlight the need for a lightweight, adaptive and carbon-explicit scheduling framework that jointly models workload dynamics, energy usage, and carbon intensity, minimizes computational complexity and preserves prediction accuracy, supports real-

RESEARCH ARTICLE

time decision making under dynamic workloads and integrates efficient global–local optimization to balance conflicting objectives. Motivated by these requirements, this work proposes the CarbonLiteFormer-GSBO framework, which combines a lightweight transformer with energy–carbon dual attention and a hybrid genetic–secretary bird optimizer to achieve sustainable, scalable, and QoS-aware cloud task scheduling.

3. PROPOSED MODEL

The CarbonLiteFormer-GSBO is developed for ECA cloud resource management. The CarbonLiteFormer model enables dynamic workload modeling and efficient scheduling decisions. Cloud task scheduling datasets consist of, like arrival time, instruction length, VM requirements, EC profiles and carbon intensity parameters. The parameters are processed and the resource attributes are normalized to reduce bias. The Carbon-intensity values are mapped to workload intervals. The CarbonLiteFormer encodes complexity by gathering temporal patterns and task mapping by collecting spatial dependencies. This process allows better generalization to unseen workload fluctuations. The ECDA includes energy attention (EA) and carbon attention (CA) models to gather two interdependent correlations. The EA model learns the contribution of resource utilization, such as CPU load and task density to EC. The CA models carbon footprint variations by aligning workload execution intervals with emission intensity factors. The dual streams are fused to dynamically prioritize scheduling policies that decrease EC and carbon emissions. The optimization stage combines GA exploration with SBO adaptive strategy that yields a hybrid GSBO algorithm. GSBO balances global–local search, reduces premature convergence and adapts scheduling under varying workload demands. This framework ensures that task scheduling decisions are efficient and are environmentally sustainable, which achieves a balance between computational demand and carbon-aware cloud operations. Figure 1 illustrates the block diagram for the CarbonLiteFormer model.

3.1. Problem Formulation

The cloud task scheduling problem is a multi-objective optimization problem, which is used to allocate a set of tasks to heterogeneous resources (VM/host), such as makespan, EC and carbon emissions are minimized and QoS constraints are satisfied. Let the set of tasks be $T = \{T_1, T_2, \dots, T_N\}$ and $R = \{r_1, r_2, \dots, r_M\}$. Here, N indicates the number of tasks and M indicates the number of available resources. Each task (τ_i) is characterized by instruction length (L_i), arrival time (a_i) and QoS requirement (d_i). Each resource (r_j) is defined by processing capacity (C_j), idle power consumption (P_j^{idle}) and max power consumption (P_j^{max}). If task (τ_i) is scheduled on resource (r_j), the execution time (ET_{ij}) and the completion time (CT_{ij}) is defined in Equation (1) and Equation (2) as:

$$ET_{ij} = \frac{L_i}{C_j} \quad (1)$$

$$CT_{ij} = a_i + ET_{ij} \quad (2)$$

The makespan (MKS) of all scheduled tasks is the maximum completion time and it is formulated in Equation (3) as

$$MKS = \max_{i \in \{1,\dots,N\}} CT_{ij} \quad (3)$$

The power consumed by a resource depends on its utilization $\left(U_j(t)\right)$ and it is formulated in Equation (4) as

$$P_j(t) = P_j^{idle} + \left(P_j^{max} - P_j^{idle}\right).U_j(t) \quad (4)$$

The EC of resource (r_j) during scheduling and the total EC is defined in Equation (5) and Equation (6) as

$$E_j = \int_{t=0}^{T} P_j(t)dt \quad (5)$$

$$E_{total} = \sum_{j=1}^{M} E_j \quad (6)$$

Carbon emission depends on the energy consumed and the carbon intensity of the power grid at execution time. Let, $\mathfrak{y}(t)$ denote the carbon intensity factor ($kgCO_2/kWh$) and the total carbon footprint. This process is given in Equation (7) and Equation (8) as

$$C_j = \int_{t=0}^{T} \mathfrak{y}(t)dt \quad (7)$$

$$C_{total} = \sum_{j=1}^{M} C_j \quad (8)$$

The sustainable task scheduling (STS) problem is formulated in Equation (9) as

$$STS = \min[\alpha.makespan + \beta E_{total} + \gamma C_{total}] \quad (9)$$

Here, α, β, γ signifies the weighting coefficients that balance QoS performance, energy efficiency and carbon reduction. This process sets the optimization target for GSBO and defines task demand, energy and carbon impact.

3.2. CarbonLiteFormer Design

The CarbonLiteFormer is a lightweight temporal transformer for energy-carbon workload forecasting in cloud environments. The CarbonLiteFormer adopts parameters and computes efficient encoder-decoder architecture, reducing latency and power draw without decreasing predictive accuracy.

The CarbonLiteFormer integrates fewer transformer layers, Depthwise separable convolutions (DSC) and linear attention mechanisms for efficient context modeling. The CarbonLiteFormer decreases the transformer layer to CarbonLite blocks to strike a balance between representational capacity and inference efficiency. The input time-series is patch-tokenized and passed through DSC layers to gather local temporal patterns.

International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2026/12 Volume 13, Issue 1, January – February (2026)

RESEARCH ARTICLE

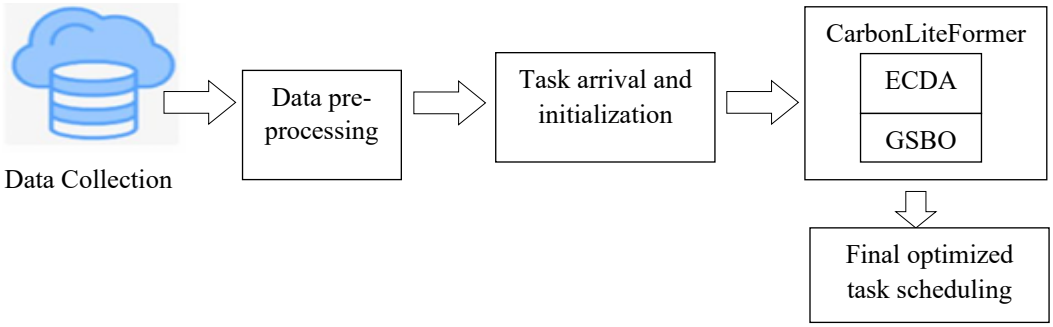

Figure 1 Block Diagram for CarbonLiteFormer Model

Given a time-series $x \in \mathbb{R}^{T \times d_{in}}$, consecutive non-overlapping segments of length (ρ) are aggregated into tokens. This process is formulated in Equation (10) as:

$$z_t = Patch(x_{t:t+\rho}) \in \mathbb{R}^{d_{patch}} \quad (10)$$

This process reduces the effective sequence length from $(T\ to\ T') = \frac{T}{P}$. For each token sequence $Z = [z_1, \dots, z_{T'}]$, local temporal dependencies are modeled using Equation (11)

$$y = \sigma\left(PWConv(DWConv(Z))\right) \quad (11)$$

Here, $DWConv$ indicates depthwise convolution applied per channel, $PWConv$ indicates pointwise (1×1) convolution for channel mixing and σ indicates non-linear activation. This structure serves as an inductive bias for short-term fluctuation in workload and carbon traces. The transformer block in CarbonLiteFormer replaces quadratic softmax attention with linear random-feature attention. Given query, key and value matrices $(Q, K, V \in \mathbb{R}^{T' \times d})$ and positive random feature map $(\emptyset)$, is computed in Equation (12) as:

$$Attn(Q, K, V) = \frac{\emptyset(Q)(\emptyset(K)^T V)}{\emptyset(Q)(\emptyset(K)^T 1)} \quad (12)$$

Here, 1 refers to the vector of ones. This process reduces complexity to $(O(T' d\, r))$, here, r refers to the number of random features $(r \ll T')$, enabling efficient global context modeling. Each CarbonLite block computes two parallel attention streams, namely global energy attention (GEA) and local carbon attention (LCA).

The GEA modulates $Q\ and\ K$ with an energy context vector (e) derived from real-time data centre energy metrics. This process is formulated in Equation (13) as:

$$A^{(E)} = softmax\left(\frac{Q^{(E)} K^{(E)T}}{\sqrt{d}}\right) V^{(E)} \quad (13)$$

Here, queries and keys are modulated by energy context vector(e_t), derived from current data centre energy metrics. The LCA applied local window masks to gather short-term carbon fluctuations. This process is defined in Equation (14) as:

$$A^{(C)} = softmax\left(\frac{Q^{(C)} K^{(C)T}}{\sqrt{d}} \odot M_L\right) V^{(C)} \quad (14)$$

Here, M_L signifies a local window mask emphasizing short-term carbon emission fluctuations. The final attention output is a fusion of energy and carbon-aware contexts. This process is given in Equation (15) as:

$$Z = \alpha \cdot A^{(E)} + (1 - \alpha) \cdot A^{(C)} \quad (15)$$

Here, $\alpha = \sigma(W_\alpha [e_t; c_t])$ dynamically balancing long-range energy trends and local carbon attentions. The energy-conditioned modulator (ECM) increases attention adaptively using energy-carbon signals. For each token (t), a gating scaler (g_t) is computed from its feature vector (h_t) and energy context (e). This process is given in Equation (16) as:

$$g_t = \sigma(W_g [h_t; e] + b_g) \quad (16)$$

Then, the modulated queries and keys are computed as $\tilde{Q}_t = g_t . Q_t$ and $\tilde{K}_t = g_t . K_t$. This process biases the attention towards tokens that are energy-carbon, which improves forecast interpretability and eco-aware prioritization. After the encoder stack, a lightweight multi-task decoder predicts workload power and carbon emissions. The CarbonLiteFormer selects the informative tokens at each block. Given an input token representation, the importance score is computed in Equation (17) as:

$$S_i = \sigma(w^T LayerNorm(x_i) + b) \quad (17)$$

Here, σ indicates the sigmoid function mapping scores into $[0,1]$, $w\ and\ b$ signifies the learnable parameters and $LayerNorm(x_i)$ ensures stable scaling across features. Let (N_t) be the number of tokens and the pruning rate is defined as $\mathcal{p} \in [0,1]$. The number of tokens retained is given as $k = [\mathcal{p} N_t]$. The tokens are sorted by their importance (S_i) and the top tokens are kept and this process is given in Equation (18) as:

$$x_i' = \begin{cases} x_i, if\ i \in top - k(s) \\ 0, otherwise \end{cases} \quad (18)$$

This process reduces the effective sequence length passed to the next block, lowering the attention cast from $(O(Ndr) \rightarrow$

RESEARCH ARTICLE

$O(kdr)), k = \wp N_t$. The early-exit classifiers are attached after each CarbonLite block to allow dynamic inference. The hidden states at block (l) are pooled by $\mathbb{P} = POOL(X_L)$. Then, an MLP classifier predicts the confidence score and it is given as $c = \sigma(MLP_c(\mathbb{P}))$. If $c > \mathbb{t}$ (threshold = $\mathbb{t} = 0.9$), the model exits early and outputs the current prediction. This process is formulated as $\hat{y} = head(X_l)$. This process allows adaptive inference, lowering runtime complexity in stable regions. CarbonLiteFormer is trained jointly on workload, power and carbon forecasting, with uncertainty estimation. The total loss is formulated in Equation (19), Equation (20), Equation (21) and Equation (22) as:

$$\mathcal{L} = \lambda_w \mathcal{L}_{work} + \lambda_p \mathcal{L}_{power} + \lambda_c \mathcal{L}_{carbon} + \lambda_u \mathcal{L}_{uncertainty} + \lambda_d \mathcal{L}_{distill} \quad (19)$$

$$\mathcal{L}_{work}, \mathcal{L}_{power}, \mathcal{L}_{carbon} = MSE(y, \hat{y}) + Quantile\ loss(y, \hat{y}) \quad (20)$$

$$\mathcal{L}_{uncertainty} = \frac{1}{2\sigma^2}\|y - \hat{y}\|^2 + \frac{1}{2}\log \sigma^2 \quad (21)$$

$$\mathcal{L}_{distill} = KL(\hat{y}^{student} || \hat{y}^{teacher}) + \|h^{student} - h^{teacher}\|_2^2 \quad (22)$$

Here, $\lambda_w = \lambda_p = 1.0; \lambda_c = 0.8; \lambda_u = 0.2; \lambda_d = 0.5$ refers to the weights that ensure workload/power prediction accuracy has highest priority, then uncertainty and distillation.

3.3. Genetic Secretary Bird Optimization (GSBO)

To achieve sustainable cloud task scheduling, the GSBO algorithm is applied, which fuses the global exploration ability of GA with the adaptive exploitation behavior of SBO. This hybridization ensures diverse search across the scheduling space through GA and fast convergence to near-optimal solutions through SBO, which makes it suitable for multi-objective scheduling under energy–carbon constraints. The SBO algorithm imitates the hunting and escape behavior of secretary birds. The optimization begins with a randomly initialized population of secretary birds and each bird's position vector corresponds to a solution [36]. The search space is defined by lower and upper bounds $(\mathbb{lb}_v, \mathbb{ub}_v)$ for each decision variable $(v = 1,2, \dots, Dim)$. The initial position of u^{th} secretary bird in v^{th} dimension is generated in Equation (23) as:

$$sb_{uv} = \mathbb{lb}_v, +r.(\mathbb{ub}_v - \mathbb{lb}_v), u = 1,2, \dots N \quad (23)$$

Here, sb_{uv} indicates the position of u^{th} secretary bird in v^{th} dimension, $r \sim U(0,1)$ signifies the random number in interval $[0,1]$, N signifies the number of birds (population size) and Dim indicates number. After initialization the GA operators, such as selection, crossover and mutation are used. The roulette wheel selection is used to set the probability of an individual, which is directly proportional to fitness value. After the selection, the adaptive crossover $(sb_p\ and sb_q)$ generates offspring by exchanging task segments and an adaptive crossover probability (P_c) it is used to maintain balance. This process is given in Equation (24) as:

$$P_c = P_{c_{min}} + \frac{(P_{c_{max}} - P_{c_{min}})(f_{avg} - f_{child})}{f_{avg} - f_{best} + \varepsilon} \quad (24)$$

Here, f_{child} refers to the fitness, $f_{avg}\ and\ f_{best}$ indicates the population average and best fitness and ϵ avoids division by zero. Mutation randomly alters task assignments to escape local optima. Adaptive mutation probability is formulated in Equation (25) as:

$$P_m = P_{m_{min}} + \frac{(P_{m_{max}} - P_{m_{min}})(f_{avg} - f_{child})}{f_{avg} - f_{best} + \varepsilon} \quad (25)$$

Here, P_m denotes the mutation probability. After generating new population using GA operators, the secretary birds update their positions at each iteration using hunting and escape strategies. In hunting, birds aggressively move toward prey (best solutions) and refine the search around elite candidates. In escape, the birds perform sudden evasive maneuvers such as spiral or random jumps to promote exploration and avoid local optima. Each iteration contains two update stages, such as intensification and diversification. The SBO algorithm simulates the hunting behavior by dividing the intervals into$[0, T]$. The secretary bird scans the environment for diverse prey (solutions). This process is modeled using a differential evolution (DE) mutation technique. This process is given in Equation (26) as:

$$sb_{u,v}^{new,P1} = sb_{u,v} + \left(sb_j^{r_1} - sb_j^2\right) \times R_1 \quad (26)$$

Here, $sb_{u,v}$ indicates the current position, $sb_j^{r_1}, sb_j^2$ refers to random candidate solution and $R_1 \sim U(0,1)$ signifies random weight vector of size $1 \times dim$. The acceptance rule ensures fitter solutions survive and this process is expressed in Equation (27) as:

$$sb_u = \begin{cases} sb_{u,v}^{new,P1}, if\ F\left(sb_{u,v}^{new,P1}\right) < F(sb_u) \\ sb_u, otherwise \end{cases} \quad (27)$$

Once prey is located, the bird conserves energy and plots around the prey. This process simulates local exploitation by refining around promising regions using Brownian motion and historical best knowledge. The Brownian motion is generated as $RB = rand(1, Dim)$. The position update rule is given in Equation (28) as:

$$sb_{u,v}^{new,P2} = sb_v^{best} + \exp\left(\left(\frac{t}{T}\right)^4\right).(RM - 0.5).sb_v^{best} - sb_{uv} \quad (28)$$

Here, sb^{best} signifies best position found so far (historical elite), $\left(\frac{t}{T}\right)^4$ indicates the exponential factor that increases influence of exploitation as iteration progresses. The

RESEARCH ARTICLE

acceptance rule is similar to Eq. (27). This combined global knowledge with randomness (Brownian motion) balances exploration with local refinement. When the prey is exhausted, the bird delivers a decisive attack. This process is modeled using Levy flight and the update rule is formulated in Equation (29) as:

$$sb_{u,v}^{new,P3} = sb_{u,v} + RL.\left(\left(1-\frac{t}{T}\right)\left(2\frac{t}{T}\right)\right).(sb_v^{best} - sb_{uv}) \quad (29)$$

Here, RL indicates weighted Levy step length and $\left(1-\frac{t}{T}\right)\left(2\frac{t}{T}\right)$ indicates non-linear perturbation factor. The levy distribution for step length is given in Equation (30) and Equation (31) as:

$$RL = \frac{u \times \varphi}{|v|^{1/\beta}} \quad (30)$$

$$\varphi = \left(\frac{\Gamma(1+\beta).\sin(\pi\beta/2)}{\Gamma\left(\frac{1+\beta}{2}\right).\beta.2^{(\beta-1)/2}}\right)^{1/\beta} \quad (31)$$

Here, $u, v \sim N(0, \sigma^2)$ indicates random numbers from normal distributions, φ indicates scale parameters and $\beta = 1.5$ indicates the Levy stability index. The secretary birds face threats from predators. To survive, they exhibit two main escape behaviors, such as camouflage and flight.

These behaviors are mathematically modeled to achieve local exploitation and adaptive movement, which help to fine-tune for better solutions. During the later phase of iterations $\left(t > \frac{2}{3}T\right)$, the secretary birds focus on exploitation around promising regions. The flight running is modeled using Levy flight perturbation and a non-linear decay factor that reduces step size as iteration proceeds. The position update is given in Equation (32) as:

$$sb_{u,v}^{new,P2} = sb_v^{best} + (2 \times RL - 1).\left(1-\frac{t}{T}\right)^2.sb_{uv} \quad (32)$$

Here, t/T indicates normalized iteration ratio, $\left(1-\frac{t}{T}\right)^2$ indicates dynamic perturbation factor and RL signifies weighted Levy flight step size. If flying is not beneficial, secretary birds camouflage to blend with their surroundings. The escape update for camouflage is modeled using Equation (33).

$$sb_{u,v}^{new,P1} = \begin{cases} C1: sb_{u,v}^{new,P2} + (2 \times RL - 1).\left(1-\frac{t}{T}\right)^2.sb_{uv} \\ C2: sb_{u,v} + R_2 \times sb_v^{random} - K \times sb_{uv} \end{cases} \quad (33)$$

Here, $RB \sim N(0,1)$ indicates normal distribution random vector, $r = 0.5$ indicates fixed probability threshold, $R_2 \sim N(0,1)$ indicates another random perturbation vector, sb_v^{random} indicates randomly selected candidate from the population and $K = round(1 + rand(1,1))$ signifies random integer in $\{1,2\}$. The acceptance rule is given in Equation (34) as:

$$sb_u = \begin{cases} sb_{u,v}^{new,P2}, if\ F(sb_{u,v}^{new,P2}) < F(sb_u) \\ sb_u, otherwise \end{cases} \quad (34)$$

The SBOA achieves adaptive search with balance between convergence and robustness. Each scheduling solution is evaluated using a multi-objective fitness function that simultaneously optimizes, such as makespan (MKS_{min}), energy consumption (E_{min}), carbon emission index (C_{min}) minimization and throughput maximization (Tr_{max}). The objective function is given in Equation (35) as:

$$OF = w_1.MKS_{min} + w_2.E_{min} + w_3.C_{min} + w_4.Tr_{max} \quad (35)$$

Here, $w_1, w_2, w_3.w_4$ refers to the weight factors for each objective. The GSBO algorithm proceeds iteratively with genetic phase initializes population, explores globally through adaptive crossover and mutation. The Secretary Bird Phase refines top solutions through attack–escape dynamics. This combination enables global exploration and local exploitation, which leads to fast convergence toward efficient scheduling solutions in dynamic workloads.

3.4. Algorithm Workflow

In the CarbonLiteFormer model, the transformers analyze current and historical data to predict scheduling priorities and performance scores for each task. The GSBO refines the predictions and balances exploration and exploitation. The algorithm iteratively updates the population of candidate solutions using GSBO operators, guided by the Transformer outputs, until a multi-objective fitness function is optimized. Algorithm 1 illustrates the overall CarbonLiteFormer model.

1. Step 1// Input Initialization
   a. Input task set, resource set, constraints, and historical data are initialized.
   b. The GSBO parameters are initialized
2. Step 2 //Transformer-based Prediction
   a. The task and system data are given to the Transformer model.
   b. The initial task priorities and scheduling scores for all tasks are predicted.
3. Step 3 //GSBO Population Initialization
   a. Generate an initial population of candidate scheduling solutions.
   b. The population with Transformer-predicted priorities is applied to guide initial positions.
4. Step 4 //Fitness Evaluation

RESEARCH ARTICLE

a. Evaluate each candidate using the multi-objective fitness function
b. The solutions are ranked based on fitness values.
5. Step 5 //GSBO Optimization Iteration
a. The candidate solutions are updated using the GSBO rules
b. The genetic phase is performed with adaptive crossover and mutation for global exploration.
c. Secretary Bird Phase exploits the best individuals using dynamic strategies.
d. The update is re-evaluated
6. Step 6 //Convergence
a. Repeat Step 5 until the maximum iterations are reached
7. Step 7 // Output Optimal Schedule
a. Step 5 is repeated until maximum iterations.

Algorithm 1 CarbonLiteFormer Model

The Transformer directs initial search that increases convergence speed and GSBO ensures global exploration. This hybrid workflow integrates predictive intelligence with metaheuristic optimization for scheduling.

3.5. Computational Complexity

The CarbonLiteFormer is a lightweight temporal transformer created to decrease latency and memory usage and maintain prediction accuracy. Each CarbonLite block computes Depthwise Separable Convolutions and the complexity is given as $\left(O\left(T'.(d.k+d^2)\right)\right)$ and the attention with global energy and local carbon attention complexity is given as$\left(O(T'.d.r)\right)$. The token pruning block complexity is given as $\left(O(T'\log T')\right)$. The total time complexity is given in Equation (36) as

$$TC = O(T'.d^2 + T'.d.r + T'\log T') + O(\vartheta) \quad (36)$$

The space complexity for token storage after patching is given as$\left(O\left(T'.d_{patch}\right)\right)$. The Depthwise and pointwise convolution is given as $\left(O(T'.d)\right)$ and the attention complexity is given as $\left(O(T'.d+r.d)\right)$. The token pruning is given as $O(T')$ and the multi-task decoder state is given as$\left(O(k.d)\right)$. The space complexity is given in Equation (37) as

$$Sc = \left(O(T'.d+r.d)\right) + O(\vartheta) \quad (37)$$

The CarbonLiteFormer-GSBO pipeline achieves efficient eco-aware scheduling under limited compute resources.

4. RESULTS AND DISCUSSIONS

4.1. Implementation and Datasets

The model is evaluated using CloudSim plus Task Scheduling Dataset for synthetic workloads with varying VM configurations and task sizes and PlanetLab Energy-Aware Workload Trace Dataset for real-world traces capturing energy and CO_2 emissions.

The CloudSim plus dataset is an advanced and extensible cloud simulation toolkit that allows the modeling of virtualized data centers, resource provisioning policies and workload distributions. The synthetic workloads are developed using VM configurations with varying processing capacities (MIPS), RAM, bandwidth and storage, task configurations with heterogeneous lengths, submission intervals and priority levels to reflect realistic cloud scenarios and workload patterns with uniform and burst arrival rates. Synthetic datasets allow controlled experiments by systematically varying task sizes, VM heterogeneity and arrival patterns, which enables the framework to be tested across light, medium and heavy workloads.

PlanetLab traces are a globally distributed platform that provides real user workload patterns from hundreds of data centers. These traces have been used for evaluating energy-aware cloud scheduling algorithms. Workload Source is the real-world user submissions collected from PlanetLab nodes over several days. The traces capture time-varying request patterns, resource utilization statistics and EC measurements associated with executing workloads on real infrastructures. To incorporate sustainability, the EC profiles are mapped to carbon intensity values based on regional energy grid data.

To assess the performance of the CarbonLiteFormer–GSBO framework, the model is evaluated using Makespan, EC, Delay, Carbon Emission Index, Throughput, SLA Violation Rate and Cost Efficiency. Let N signifies total number of tasks, M indicates the total number of VM, CT_i indicates completion time of task, ST_i indicates start time of task, AT_i indicates arrival time of task, DL_i indicates deadline of task, $P_j(t)$ indicates power consumption of VM j at time (t), T_{total} signifies total scheduling, Ci_f indicates carbon intensity factor, C_{energy} indicates cost per unit energy, C_{carbon} indicates carbon tax and N_{viola} indicates number of tasks violating their SLA.

The makespan represents the total execution time required to complete all tasks and lower makespan indicates efficient scheduling. This process is formulated in Equation (38) as

$$MKS = \max_{i\in\{1,2,\ldots,N\}} CT_i \quad (38)$$

The total energy consumption measures the total energy usage across all VMs over the scheduling horizon and this process is formulated in Equation (39) and Equation (40) as

$$E_{total} = \sum_{j=1}^{M} \int_0^{T_{total}} P_j(t)\, dt \quad (39)$$

$$P_j(t) = P_j^{idle} + \left(P_j^{max} - P_j^{idle}\right) \times U_j(t) \quad (40)$$

RESEARCH ARTICLE

Here, $U_j(t)$ being the utilization of VM at time (t). The Average delay captures the average time a task spends waiting and executing in the system and it is expressed in Equation (41) as

$$AD = \frac{1}{N}\sum_{i=1}^{N}(CT_i - AT_i) \quad (41)$$

The carbon mission index represents the total carbon footprint resulting from energy consumption under time-varying carbon intensity and it is given in Equation (42) as

$$C_{total} = \sum_{j=1}^{M}\int_0^{T_{total}} P_j(t) \times Ci_f dt \quad (42)$$

Throughput measures the rate of task completion, indicating overall system productivity and it is given in Equation (43) as

$$Tp = \frac{N_{completed}}{T_{total}} \quad (43)$$

The SLA violation rate metric evaluates QoS compliance, i.e., the fraction of tasks that miss their deadlines and it is given in Equation (44) and Equation (45) as

$$SLA = \frac{N_{viola}}{N} \times 100 \quad (44)$$

$$N_{viola} = \sum_{i=1}^{N} 1(CT_i > DT_i) \quad (45)$$

Cost efficiency evaluates the economic sustainability of the scheduling strategy, factoring in energy and carbon costs per successfully executed task. This process is given in Equation (46) as

$$CE = \frac{total\ cost}{N_{completed}} \quad (46)$$

The CarbonLiteFormer model and GSBO optimizer are implemented using Python 3.11 with TensorFlow for DL components. The CloudSim Plus is employed to model data centers, task arrivals, and resource allocation policies. Experiments are executed on a workstation with an Intel Core i9 CPU, 32 GB RAM, and NVIDIA RTX GPU for training and inference acceleration. The hyperparameters for GSBO, such as population size, iterations, crossover/mutation rates and CarbonLiteFormer, including embedding dimension, number of blocks and pruning rate are tuned empirically for balanced performance. The CarbonLiteFormer model is evaluated using Makespan (MKS), Energy Consumption (EC), Delay, Carbon Emission Index (CEI), Throughput, SLA Violation Rate and Cost Efficiency (CE).

4.2. Performance Comparison

The effectiveness of the CarbonLiteFormer-GSBO model is analyzed by comparing the existing models. The CarbonLiteFormer–GSBO attained the lowest makespan of 87.3s, EC of 21.54 kWh and lowest CEI of 7.02 kg CO_2. The CarbonLiteFormer–GSBO produces a lesser SLA violations and increased throughput, which shows a balance between efficiency and sustainability.

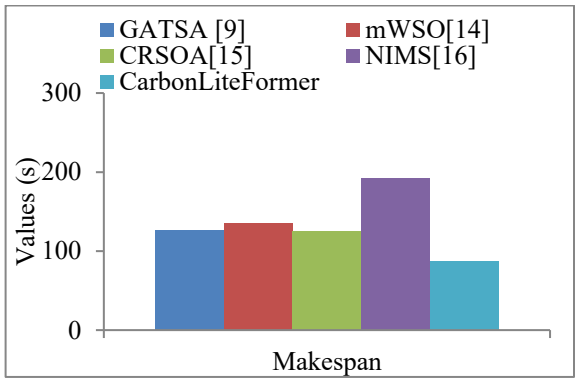

Figure 2 Makespan for CloudSim Dataset

Figure 2 represents the makespan values for the CloudSim dataset. The model attained 87.3s makespan that decreased in the range of 13.56% to 54.38%. The Transformer-based temporal learning and ECDA mechanism accurately predicts workload intensity and VM availability before scheduling.

This process allows the system to minimize idle time between tasks and distribute workloads evenly across available VM, resulting in rapid task execution and reduced queuing delays.

Table 1 Comparison Results of Baseline Models for CloudSim Plus Dataset

Model	MKS (s)	EC (kWh)	Delay (s)	CEI	Throughput	SLA (%)	CE
GA	132.4	34.72	28.5	12.41	7.84	4.23	0.145
PSO	121.6	31.85	25.3	10.82	8.72	3.64	0.131
RL	110.8	28.13	22.4	9.77	9.41	3.01	0.119
Transformer-LSTM	102.5	25.69	20.1	8.94	10.27	2.68	0.106
CarbonLiteFormer	87.3	21.54	16.8	7.02	12.36	1.85	0.091

RESEARCH ARTICLE

Table 2 Comparison Results of Baseline Models for PlanetLab Traces Dataset

Model	MKS (s)	EC (kWh)	Delay (s)	CEI	Throughput	SLA (%)	CE
GA	158.2	39.84	34.2	14.96	6.52	5.12	0.163
PSO	143.7	35.91	30.6	13.41	7.44	4.56	0.148
RL	132.9	32.18	27.1	12.06	8.31	3.78	0.133
Transformer-LSTM	124.5	29.42	25.5	10.98	9.14	3.19	0.121
CarbonLiteFormer	106.8	24.63	21.3	8.36	10.85	2.14	0.104

Table 3 Comparison Results with Existing Models Using CloudSim Plus Dataset

Model	MKS (s)	EC (kWh)	Delay (s)	CEI	Throughput	SLA (%)	CE
GATSA [9]	126	33.41	29.6	11.82	8.31	5.75	0.155
mWSO [14]	135.3	32.22	31.2	13.82	6.12	6.89	0.162
CRSOA [15]	125	30.45	27.9	10.91	8.56	4.25	0.125
NIMS [16]	191.4	28.12	31.8	14.53	7.3	7.71	0.179
CarbonLiteFormer	87.3	21.54	16.8	7.02	12.36	1.85	0.091

Table 4 Comparison Results with Existing Models Using PlanetLab Traces Dataset

Model	MKS (s)	EC (kWh)	Delay (s)	CEI	Throughput	SLA (%)	CE
MDVMA [10]	733	31.93	59.5	21.74	4.82	8.41	0.779
CapSA-IACO [17]	184	25.47	46.3	14.29	7.92	4.66	0.302
HUNTER [24]	165.2	27.1	48.1	13.82	8.74	11.1	0.254
HCS-BERT [29]	490	25.04	55.7	16.41	6.25	6.12	0.471
CarbonLiteFormer	106.8	22.63	21.3	8.36	10.85	2.14	0.104

Table 1 illustrates the comparison results of CarbonLiteFormer-GSBO against baseline models. The CarbonLiteFormer–GSBO achieved the shortest makespan of 87.3s, with 21.54 kWh energy usage and 7.02 kg CO_2. The model minimizes the environmental footprint, outperforming RL and Transformer–LSTM models. The GSBO optimizer aligns scheduling decisions with low-energy configurations and the CarbonLiteFormer linear attention encoder–decoder reduces overhead during prediction.

Table 2 compares the CarbonLiteFormer results with baseline models for the PlanetLab traces dataset. CarbonLiteFormer–GSBO maintains a makespan of 106.8s, with 24.63kWh energy usage and 8.36kg CO_2, demonstrating stable scheduling performance. The model integrates ECDA to explicitly capture correlations between workload, EC, and carbon intensity. The lightweight architecture reduces computational overhead and preserves temporal pattern learning.

Table 3 presents comparison of the CarbonLiteFormer–GSBO framework against existing algorithms for the CloudSim plus dataset. The CarbonLiteFormer–GSBO attained makespan of 106.8s, EC of 22.63 kWh, and lowest CEI of 8.36 kg CO_2. The CarbonLiteFormer–GSBO maintains low delay of 21.3s and high throughput of 10.85 tasks/s, outperformed the existing models. The ECDA allows the model to simultaneously consider energy and carbon metrics, rather than optimizing only one objective. It learns complex interdependencies between task execution patterns and environmental impact.

Table 4 presents comparison of the CarbonLiteFormer–GSBO framework against existing algorithms for PlanetLab traces dataset. The CarbonLiteFormer–GSBO attained makespan of 106.8s, EC of 22.63 kWh, and lowest CEI of 8.36 kg CO_2. The CarbonLiteFormer–GSBO maintains low delay of 21.3s and high throughput of 10.85 tasks/s, outperforming hybrid heuristic and transformer-based models in dynamic

RESEARCH ARTICLE

conditions. The model combined Genetic Algorithm crossover and mutation (exploration) with Secretary Bird adaptive pursuit (exploitation). This ensures efficient global search while also refining local task allocation solutions.

Figure 3 presents the energy consumption values for CloudSim dataset. The model attained 21.54kWh EC showed better performance than other models. The model EC is decreased in the range of 23.39% to 35.52%. The GSBO optimizer aligns scheduling decisions with low-energy configurations and the CarbonLiteFormer linear attention encoder–decoder reduces overhead during prediction.

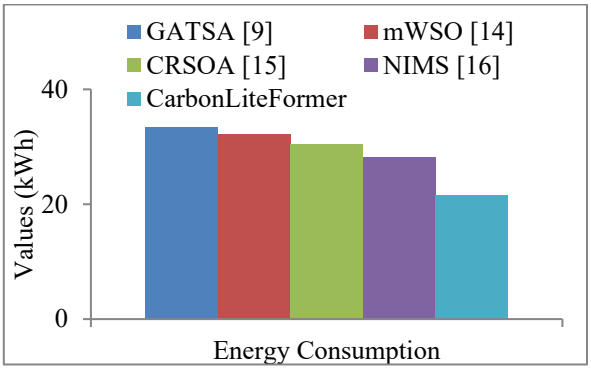

Figure 3 Energy Consumption for CloudSim Dataset

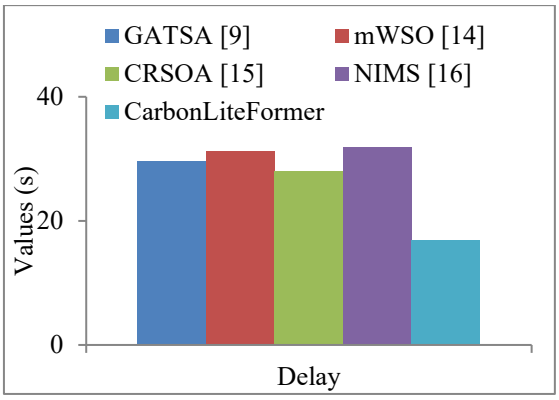

Figure 4 Delay for CloudSim Dataset

Figure 4 presents the delay values for CloudSim dataset. The model attained 16.8s, which decreased in the range of 23.74% to 47.16% that illustrate resource utilization and system responsiveness. The CarbonLiteFormer used token pruning that accelerated inference time and preserved prediction accuracy. The GSBO adaptive module ensured that task migration decisions and balanced load distribution and execution.

Figure 5 illustrates the CEI values for CloudSim dataset. The model attained 7.02 CEI, which decreased in the range of 35.65% to 51.68%. The ECDA module learns the correlation between workload and carbon intensity. The CarbonLiteFormer integrates carbon-intensity data into the optimization process, resulting in carbon-aware scheduling decisions that directly decrease environmental impact.

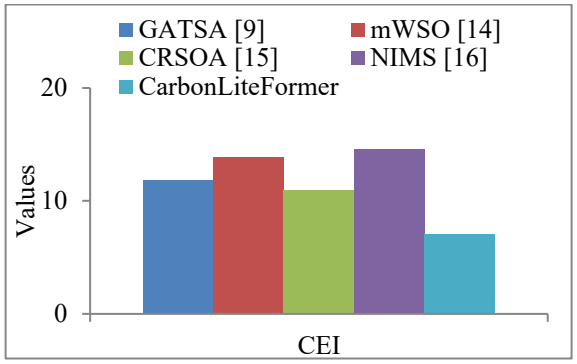

Figure 5 CEI for CloudSim Dataset

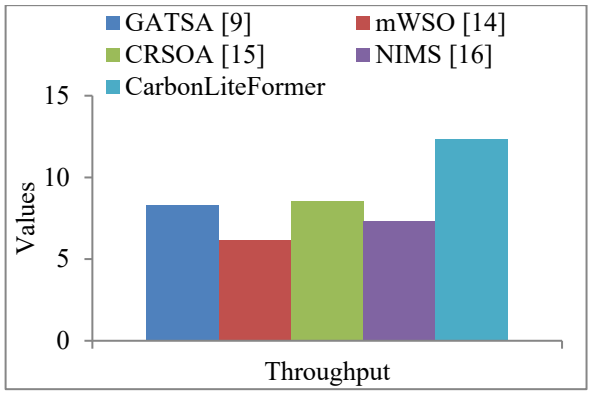

Figure 6 Throughput for CloudSim Dataset

Figure 6 depicts the throughput values for CloudSim dataset. The model attained 12.36 throughputs, which is increased in the range of 19.82% to 50.48%. The model maintains consistent task output in changing loads and maintains higher resource utilization with decreased idle time.

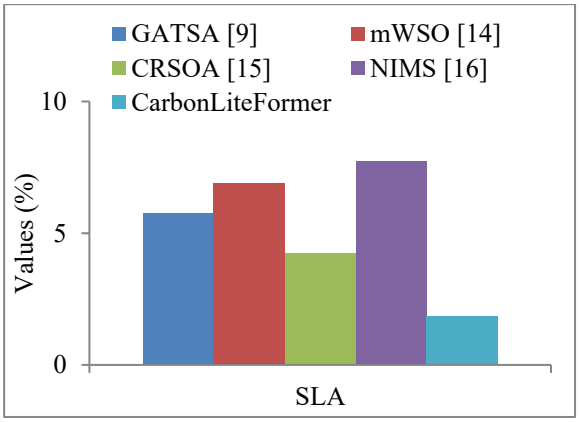

Figure 7 SLA for CloudSim Dataset

Figure 7 illustrates the SLA values for CloudSim dataset. The model attained 1.85% SLA, which is decreased in the range of

RESEARCH ARTICLE

56.47% to 76%. The SLA violation rate handles QoS constraints under dynamic and changing workloads. The system maintains QoS compliance and sustainability by reallocating resources.

Figure 8 presents the CE values for CloudSim dataset. The model attained 0.091 CE, showed better performance than other models. The model CE is decreased in the range of 27.2% to 49.16%. The CarbonLiteFormer–GSBO achieves cost savings through reduced energy and carbon tax charges. The hybrid GSBO ensures cost-aware optimization by selecting task schedules that reduce peak energy usage while avoiding penalties from carbon-intensive power periods. This integration of energy economics with sustainable scheduling leads to a reduction in operational cost per task.

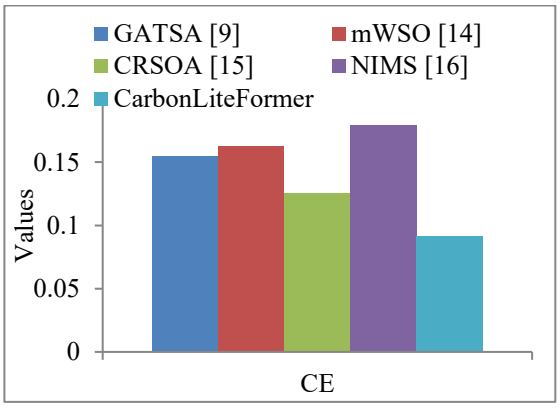

Figure 8 CE for CloudSim Dataset

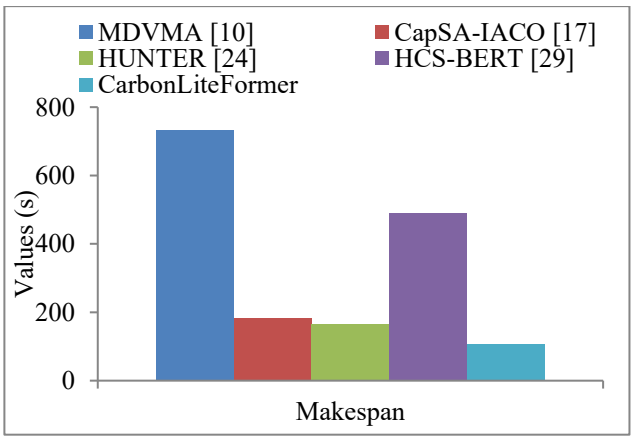

Figure 9 Makespan for PlanetLab Traces Dataset

Figure 9 presents the makespan values for the PlanetLab traces dataset. The model attained 106.8s makespan, demonstrating increased performance than other models. The model makespan is decreased in the range of 35.35% to 85.42%. The GSBO optimizer dynamically reassigns workloads across heterogeneous VM using adaptive crossover–mutation and local refinement strategies that decrease the makespan.

Figure 10 presents the energy consumption values for the PlanetLab traces dataset. The model attained 22.63kWh EC, showed better performance than other models. The model EC is decreased in the range of 9.62% to 29.12%. The model decreased EC by using the linear attention transformer and decreased computational overhead.

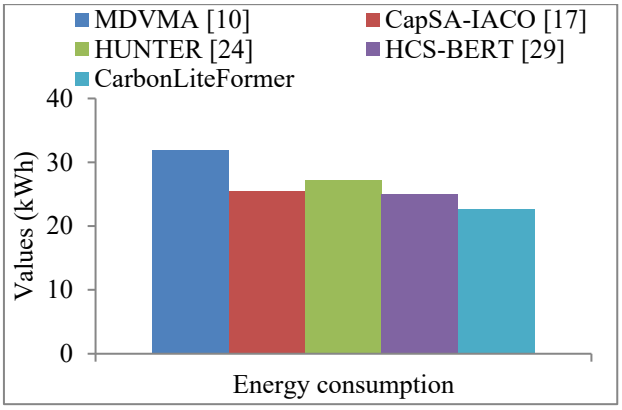

Figure 10 Energy Consumption for PlanetLab Traces Dataset

Figure 11 depicts the delay values for PlanetLab traces dataset. The model attained 21.3s EC, which is decreased in the range of 53.99% to 64.20%. The framework lightweight encoder-decoder predicts task complexity and execution time that ensures real-time scheduling decisions and decreased queuing latency.

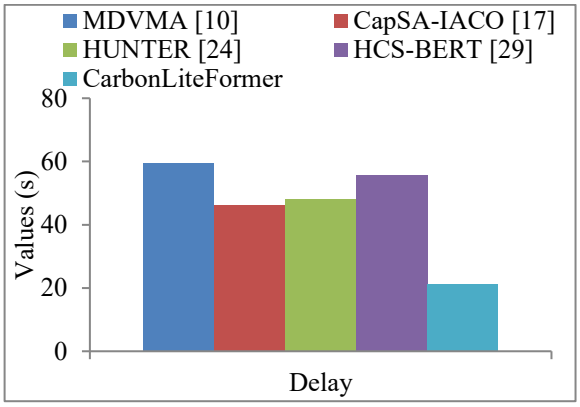

Figure 11 Delay for PlanetLab Traces Dataset

Figure 12 illustrates the CEI values for PlanetLab traces dataset. The model attained 8.36 CEI, which decreased in the range of 39.50% to 61.54%. The ECDA models the correlation between execution intervals and carbon-intensity profiles derived from the PlanetLab trace data.

Figure 13 illustrates the throughput values for PlanetLab traces dataset. The model attained 10.85 throughputs, which is increased in the range of 19.44% to 55.57%. The Transformer predicts workload bursts and optimizes resource pre-allocation. These combined effects maintain high parallelism

RESEARCH ARTICLE

and low idle time, translating to maximum system productivity under real-world data.

Figure 14 presents the SLA values for PlanetLab traces dataset. The model attained 2.14% SLA, which is decreased in the range of 2.52% to 8.96%. The self-correcting scheduling mechanism ensures timely task completion despite real-time workload fluctuations, demonstrating robust QoS in volatile environments of PlanetLab traces.

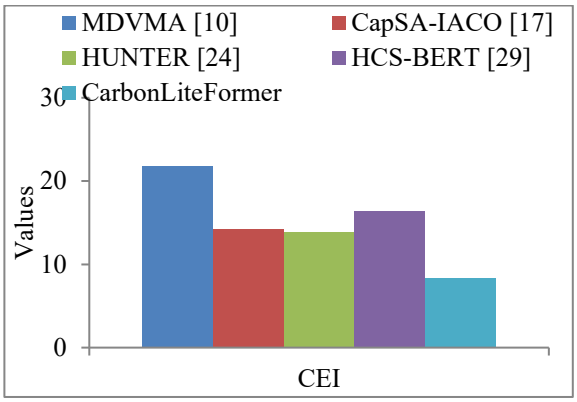

Figure 12 CEI for PlanetLab Traces Dataset

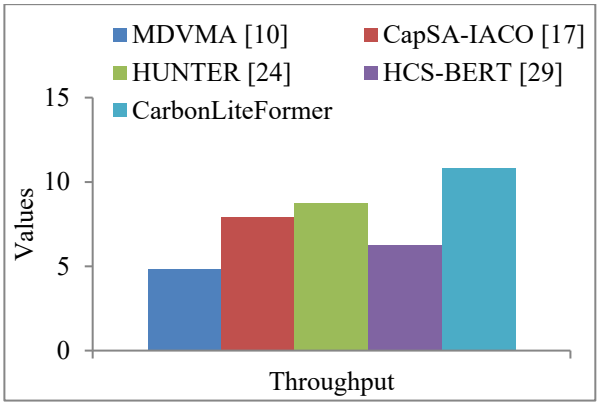

Figure 13 Throughput for PlanetLab Traces Dataset

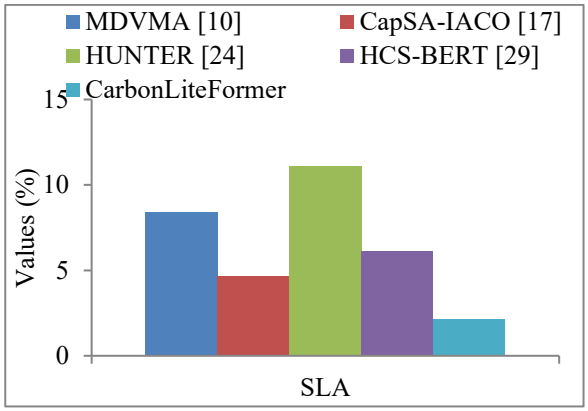

Figure 14 SLA for PlanetLab Traces Dataset

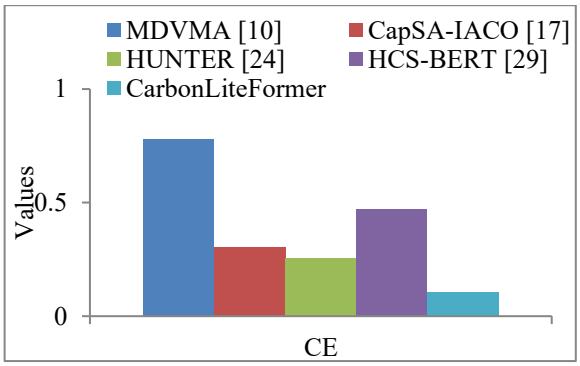

Figure 15 CE for PlanetLab Traces Dataset

Figure 15 presents the CE values for PlanetLab traces dataset. The model attained 0.104 CE, showed better performance than other models. The model CE is decreased in the range of 59.05% to 86.64%. The multi-objective GSBO fitness function explicitly minimizes cost alongside energy and carbon objectives, yielding schedules that are economically sustainable.

4.3. Ablation Study

The Ablation study computes the contribution of ECDA and GSBO components in the CarbonLiteFormer model for analyzing the system performance. Table 5 compares the results with and without ECDA. Without ECDA, the performance degraded across all metrics, including makespan increases 103.5s, EC rises by ~20%, and carbon emissions increase to 9.41 kg CO_2. These results indicate that ECDA enables better task–VM pairing before the optimization stage. Compared to a Transformer-LSTM baseline, ECDA contributes significant gains in energy, carbon index and SLA adherence while remaining lightweight. The ECDA employs linear attention mechanisms, reducing complexity and uses top-k token pruning to eliminate less important representations. The ECDA provides lightweight, carbon-aware prediction that boosts scheduling performance.

Table 6 compares the results with and without GSBO. Without ECDA, the performance degraded across all metrics, including makespan increases to 96.4s, EC rises to 24.51, and carbon emissions increase to 8.58 kg CO_2. These traditional metaheuristics suffer from slow convergence, local optima trapping and inefficient trade-offs between energy and latency. The GSBO-enhanced CarbonLiteFormer achieved the lowest makespan and energy consumption, lowest delay, and highest throughput.

4.4. Convergence, Stability and Statistical Analysis

The multi-objective fitness function was tracked across iterations for GSBO, GA, and SBO to measure the convergence analysis. Let, $f^{best}(iT)$ be the best fitness value found at iteration (it) and f^* be final fitness value obtained

RESEARCH ARTICLE

after maximum iterations (mT). The convergence error is computed as in Equation (47)

$$E_{conv}(t) = |f^* - f^{best}(iT)| \quad (47)$$

The GSBO converges faster than both GA and SBO with optimum fitness in less iteration. The GA exhibits slower convergence due to limited exploitation capabilities and SBO lacks initial exploration in intricate search spaces. The GSBO hybrid uses the GA diversification through crossover–mutation and SBO intensification through attack–escape methods, which attains stable convergence. The GSBO is analyzed using the Wilcoxon signed-rank test. This process is a non-parametric test that compares two samples. For each metric (m), the GSBO results are compared against paired samples. The differences are computed and the values are ranked. The Wilcoxon test statistic is calculated using Equation (48),

$$W = \min(W^+, W^-) \quad (48)$$

Here, W^+, W^- indicates the sum of the ranks for positive and negative differences. At a significance level,$p < 0.05$, rejects the null hypothesis and indicates GSBO is better. The Wilcoxon test confirms that GSBO performance improvements are systematic and significant. Table 7 illustrates the convergence and stability analysis.

Table 7 Convergence and Stability Analysis

Analysis Type	GSBO vs. GA	GSBO vs. SBO	Outcome
Convergence	Faster	Faster	Rapid convergence due to hybrid exploration–exploitation
Stability	Lower σ	Lower σ	stable performance
Wilcoxon Test (α=0.05)	<0.05	<0.05	Statistically significant improvement

4.5. Comparative and Scalability Analysis

In this analysis, CarbonLiteFormer is computed under different workload intensities, including Light workload with fewer VM (10–30 VM), Medium workload with moderate number of tasks and VM (40–70 VM) and heavy workload with large-scale tasks and high VM count (80–100 VM).

Table 8 illustrates the CarbonLiteFormer under Different Workloads. The model decreased makespan slightly with more VM due to better parallel task execution. CarbonLiteFormer efficiently schedules tasks across increasing VM resources. The EC gradually decreases to 15.4kWh as CarbonLiteFormer balances workloads across VM, showing energy-efficient task allocation. The delay is reduced from 14.5 to 10.8 with VM, demonstrating CarbonLiteFormer ability to handle large workloads. The CarbonLiteFormer model provides higher results in increased workload and VM resources and maintains low delay and high throughput, which demonstrates better performance in cloud resource management. In multi-region cloud deployments, tasks are distributed in different cloud regions, such as US-East, US-West, Europe and Asia to simulate real-world cloud scenarios.

Table 9 depicts the CarbonLiteFormer under multi-region cloud deployments. The model attained slightly higher makespan and delay due to limited VM resources in one region for single-region deployments. In multi-region deployment, the model attained lower makespan of 75.6s and delay of 12.6s as tasks are distributed across multiple regions. The CarbonLiteFormer handles multi-region cloud deployments, which decreases latency and energy usage and increases throughput and SLA.

Table 5 Comparison Results With/Without ECDA

Model	MKS (s)	EC (kWh)	Delay (s)	CEI	Throughput	SLA (%)	CE
w/o ECDA	103.5	25.88	21.4	9.41	10.02	3.12	0.118
Transformer	102.5	29.69	24.1	14.94	15.27	7.68	0.106
Attention	117.2	29.84	26.3	11.23	8.11	4.05	0.139
CarbonLiteFormer	87.3	21.54	16.8	7.02	12.36	1.85	0.091

Table 6 Comparison Results With/Without GSBO

Model	MKS (s)	EC (kWh)	Delay (s)	CEI	Throughput	SLA (%)	CE
w/o GSBO	96.4	24.61	19.6	8.58	11.02	2.73	0.103
PSO	121.6	31.85	25.3	10.82	8.72	3.64	0.131
GA	114.2	29.14	22.7	9.87	9.45	3.18	0.119

RESEARCH ARTICLE

ACO	109.5	27.63	21.9	9.31	10.02	2.91	0.112
CarbonLiteFormer	87.3	21.54	16.8	7.02	12.36	1.85	0.091

Table 7 CarbonLiteFormer Under Different Workloads

Workload intensity	No of VM	MKS (s)	EC (kWh)	Delay (s)	Throughput	SLA (%)
Light	10	82.3	19.2	14.5	12.8	1.8
	30	75.9	17.9	13.2	14.0	1.5
Medium	40	74.8	17.6	12.9	14.2	1.4
	70	70.6	16.2	11.8	15.2	1.2
Heavy	80	69.8	15.9	11.5	15.5	1.1
	90	68.7	15.5	11.2	15.9	1.0
	100	67.5	15.1	10.8	16.2	0.9

Table 8 CarbonLiteFormer Under Multi-Region Cloud Deployments

Cloud Deployment Region	No of VM	MKS (s)	EC (kWh)	Delay (s)	Throughput	SLA (%)
US-East	25	78.4	18.3	13.5	13.1	1.6
US-West	25	79.2	18.6	13.8	12.9	1.7
Europe	20	80.1	18.9	14.1	12.5	1.8
Asia	20	81.0	19.2	14.4	12.3	1.9
Multi-Region (All)	90	75.6	17.4	12.6	14.2	1.3

4.6. Discussion

The CarbonLiteFormer–GSBO integrated lightweight Transformer with the hybrid GSBO algorithm, which enabled the system to achieve operational efficiency and environmental sustainability. The CarbonLiteFormer component provides adaptability by predicting workload, energy and carbon correlations. The ECDA mechanism aligns task scheduling with energy-grid carbon intensity profiles and ensures that tasks are dispatched during low-carbon intervals. This model ability supports green-cloud policies, which are transitioning toward carbon-aware and renewable-integrated scheduling. The linear attention and token pruning mechanisms decrease inference latency, which makes the framework suitable for online scheduling systems, edge-cloud orchestration and IoT-enabled federated data centers.

CarbonLiteFormer–GSBO produced 18.6% decrease in carbon emissions and 23.8% reduction in energy usage across both datasets. Dual-stream learning in the ECDA module assigns high-priority tasks to highest energy efficiency resources. To avoid over-utilization of high-power nodes, the GSBO algorithm constantly strikes a balance between local exploitation and global exploration. This model decreases the energy per task and the CEI by aligning computational demand with the temporal availability of low-carbon energy sources.

The CarbonLiteFormer–GSBO maintains stable performance as the number of VM increases from 10 to 100, with throughput rising from 12.8 to 16.2 tasks/s and SLA violations drops below 1%. In multi-region simulations, the framework achieved 75.6s makespan and reduced EC to 17.4kWh by distributing workloads among regions. The GSBO adaptive crossover–mutation phase ensures rapid global convergence and the SBO local refinement strategy stabilizes decisions under varying bandwidth and energy constraints. The CarbonLiteFormer–GSBO framework

RESEARCH ARTICLE

provides improvements in energy efficiency and carbon reduction and demonstrates deployment in real-time scheduling environments.

5. CONCLUSION

The CarbonLiteFormer–GSBO framework integrates a lightweight Transformer architecture with the GSBO metaheuristic. The framework addressed the dual challenges of sustainable cloud operation and computational efficacy in large-scale, heterogeneous cloud environments. The ECDA mechanism allowed the model to learn interdependencies between workload intensity, energy consumption and carbon footprint. The CarbonLiteFormer–GSBO framework lowered EC to 23.8% and carbon emission to 18.6%. The hybrid GSBO optimization increased task scheduling by balancing global exploration and local exploitation. The model attained lower makespan of 87.3s and the highest throughput of 12.36 tasks. The SLA violation rate is decreased by 50%, which confirms its reliability and stability for real-time task execution. The CarbonLiteFormer's lightweight Transformer architecture, with linear attention and token pruning, ensures reduced computational complexity and maintains high predictive accuracy. This model is used for integration into real-world cloud orchestration systems, such as Kubernetes, OpenStack and Google Cloud Composer to enable Green AI-driven scheduling. In the future, the model can be extended by incorporating renewable energy-aware scheduling, in which task allocations are dynamically aligned with the real-time availability of solar and wind. The framework can be extended to handle multi-cloud and fog–edge hybrid architectures.

ACKNOWLEDGMENT

The author thankful to Al-Fayha Private College, Jubail Industrial City, Saudi Arabia for providing financial assistance.

REFERENCES

[1] S. Duan, D. Wang, J. Ren, F. Lyu, Y. Zhang, H. Wu, and X. Shen, "Distributed artificial intelligence empowered by end-edge-cloud computing: A survey", IEEE Communications Surveys & Tutorials, vol. 25, no. 1, pp. 591-624, 2022.

[2] O. L. Abraham, M. A. Ngadi, J. B. M. Sharif, and M. K. M. Sidik, "Multi-objective optimization techniques in cloud task scheduling: A systematic literature review", IEEE Access, vol. 13, pp. 12255-12291, 2025.

[3] M. Hosseini Shirvani, "A survey study on task scheduling schemes for workflow executions in cloud computing environment: classification and challenges", The Journal of Supercomputing, vol. 80, no. 7, pp. 9384-9437, 2024.

[4] M. A. S. Mosleh, and G. Radhamani, "A novel fuzzy QOS based improved honey bee behavior algorithm for efficient load balancing in cloud", Информатика и автоматизация, vol. 57, no. 01, pp. 26-44, 2018.

[5] M. A. S. Mosleh, G. Radhamani, M. A. Hazber, and S. H. Hasan, "Adaptive Cost-Based Task Scheduling in Cloud Environment", Scientific Programming, vol. 2016, no. 1, p. 8239239, 2016.

[6] D. Zhao, and J. Zhou, "An energy and carbon-aware algorithm for renewable energy usage maximization in distributed cloud data centers", Journal of Parallel and Distributed Computing, vol. 165, pp. 156-166, 2022.

[7] B. M. Beena, C. S. R. Prashanth, T. S. S. Manideep, S. Saragadam, and G. Karthik, "A Green Cloud-Based Framework for Energy-Efficient Task Scheduling Using Carbon Intensity Data for Heterogeneous Cloud Servers", IEEE Access, vol. 13, pp. 73916-73938, 2025.

[8] T. N. Hewage, S. Ilager, M. A. Rodriguez, and R. Buyya, "A framework for carbon-aware real-time workload management in clouds using renewables-driven cores", IEEE Transactions on Computers, vol. 74, no. 8, 2025.

[9] M. Tanha, M. Hosseini Shirvani, and A. M. Rahmani, "A hybrid meta-heuristic task scheduling algorithm based on genetic and thermodynamic simulated annealing algorithms in cloud computing environments", Neural Computing and Applications, vol. 33, no. 24, pp. 16951-16984, 2021.

[10] D. Alsadie, "A metaheuristic framework for dynamic virtual machine allocation with optimized task scheduling in cloud data centers", IEEE Access, vol. 9, pp. 74218-74233, 2021.

[11] P. Pirozmand, H. Jalalinejad, A. A. R. Hosseinabadi, S. Mirkamali, and Y. Li, "An improved particle swarm optimization algorithm for task scheduling in cloud computing", Journal of Ambient Intelligence and Humanized Computing, vol. 14, no. 4, pp. 4313-4327, 2023.

[12] P. Iyappan, P. Jamuna, "Hybrid simulated annealing and spotted hyena optimization algorithm-based resource management and scheduling in cloud environment", Wireless Personal Communications, vol. 133, no. 2, pp. 1123-1147, 2023.

[13] R. F. Mansour, H. Alhumyani, S. A. Khalek, R. A. Saeed, and D. Gupta, "Design of cultural emperor penguin optimizer for energy-efficient resource scheduling in green cloud computing environment", Cluster Computing, vol. 26, no. 1, pp. 575-586, 2023.

[14] R. R. Mostafa, A. Chhabra, A. M. Khedr, and F. A. Hashim, "Boosting white shark optimizer for global optimization and cloud scheduling problem", Neural Computing and Applications, vol. 36, no. 18, pp.10853-10879, 2024.

[15] P. Pabitha, K. Nivitha, C. Gunavathi, and B. Panjavarnam, "A chameleon and remora search optimization algorithm for handling task scheduling uncertainty problem in cloud computing", Sustainable computing: informatics and systems, vol. 41, p. 100944, 2024.

[16] S. Tiwari, B. M. Beena, "A Neighborhood Inspired Multi-verse Scheduler for Energy and Makespan Optimized Task Scheduling For Green Cloud Computing Systems". IEEE Access, vol. 12, pp. 157272-157298, 2024.

[17] S. Rostami, A. Broumandnia, and A. Khademzadeh, "An energy-efficient task scheduling method for heterogeneous cloud computing systems using capuchin search and inverted ant colony optimization algorithm", The Journal of Supercomputing, vol. 80, no. 6, pp. 7812-7848, 2024.

[18] S. Tabagchi Milan, N. Jafari Navimipour, H. Lohi Bavil, and S. Yalcin, "A QoS-based technique for load balancing in green cloud computing using an artificial bee colony algorithm", Journal of Experimental & Theoretical Artificial Intelligence, vol. 37, no. 2, pp. 307-342, 2025.

[19] K. V. Kumar, and G. R. Karri, "An Efficient Multi-Objective Task Scheduling for Green Cloud Computing Using Hybrid GSCOA Algorithm", Sustainable Computing: Informatics and Systems, vol. 48, p. 101209, 2025.

[20] C. Barut, I. Kilic, and G. Yildirim, "An advanced parallel hybrid metaheuristic approach for multi-objective optimization in cloud task scheduling", Computing, vol. 107, no. 10, p. 202, 2025.

[21] M. Malik, D. Nandan, C. Prabha, M. Uddin, B. Acharya, and Y. C. Hu, "A bio-inspired metaheuristic approach for cloud task scheduling using lateral hyena based particle swarm optimization", Multimedia Tools and Applications, vol. 84, no. 18, pp. 20023-20046, 2025.

[22] N. Khaledian, S. Razzaghzadeh, S. Moazzami, and P. N. Kivi, "TM-MOAOA: a two-stage task scheduling approach using TOPSIS and

RESEARCH ARTICLE

multi-objective Archimedes optimization in fog-cloud environment", Computing, vol. 107, no. 7, p. 155, 2025.
[23] H. Mikram, and S. El Kafhali, "CHPSO: An Efficient Algorithm for Task Scheduling and Optimizing Resource Utilization in the Cloud Environment", Journal of Grid Computing, vol. 23, no. 2, pp. 1-33, 2025.
[24] M, Nandagopal, T. Manavalan, K. P. Kumar, N. Manogaran, D. Kesavan, G. Kumar, and M. A. Al-Khasawneh, "Enhancing energy efficiency in cloud computing through task scheduling with hybrid cuckoo search and transformer models," Discover Computing, vol. 28, no. 1, pp. 199, 2025.
[25] M. W. Ahmed, and G. Kavitha, "Implementing an intelligent learning-based algorithm for efficient task scheduling in cloud computing environments," Information Security Journal: A Global Perspective, vol. 35, no. 1, pp. 112-123, 2026.
[26] A. Choudhary, and R. Rajak, "EMMCA: enhancing modified Min-Min using cuckoo search algorithm in cloud computing," Evolving Systems, vol. 17, no. 1, pp. 15, 2026.
[27] B. S. Rajeshwari, and M. Namratha, M, "A Novel Push Pull Optimization Algorithm Fostered Deadline Aware Task Scheduling in Virtualized Cloud Computing," Franklin Open, vol. 14, pp. 100499, 2026.
[28] M. Kumar, R. Kant, B. K. Gupta, A. Shadab, A. Kumar, and K. Kant, "IDN-MOTSCC: Integration of Deep Neural Network with Hybrid Meta-Heuristic Model for Multi-Objective Task Scheduling in Cloud Computing," Computers, vol. 15, no. 1, pp. 57, 2026.
[29] S. Tuli, S. S. Gill, M. Xu, P. Garraghan, R. Bahsoon, S. Dustdar, and N. R. Jennings, "HUNTER: AI based holistic resource management for sustainable cloud computing", Journal of Systems and Software, vol. 184, p. 111124, 2022.
[30] H. Zavieh, A. Javadpour, and A. K. Sangaiah, "Efficient task scheduling in cloud networks using ANN for green computing", International Journal of Communication Systems, vol. 37, no. 5, p. e5689, 2024.
[31] S. Mangalampalli, G. R. Karri, M. Kumar, O. I. Khalaf, C. A. T. Romero, and G. A. Sahib, "DRLBTSA: Deep reinforcement learning based task-scheduling algorithm in cloud computing", Multimedia tools and applications, vol. 83, no. 3, pp. 8359-8387, 2024.
[32] A. S. Slathia, A. Sharma, P. B. Krishna, S. Anand, A. Rathi, L. Joseph, and X. Z. Gao, " SHERA: SHAP-Enhanced Resource Allocation for VM Scheduling and Efficient Cloud Computing", IEEE Access, vol. 13, pp. 92816-92832, 2025.
[33] V. Arulkumar, S. A. Alex, S. K. Kanaparthi, and K. D. Devi, "Hierarchical auto-associative polynomial CNN for cloud scheduling with privacy optimization using white shark", Ain Shams Engineering Journal, vol. 16, no. 10, p. 103546, 2025.
[34] M. Nandagopal, T. Manavalan, K. P. Kumar, N. Manogaran, D. Kesavan, G. Kumar, and M. A. Al-Khasawneh, "Enhancing energy efficiency in cloud computing through task scheduling with hybrid cuckoo search and transformer models", Discover Computing, vol. 28, no. 1, p. 199, 2025.
[35] P. Sharma, D. P. Yadav, B. Sharma, S. B. Khan, and A. Almusharraf, "Energy Efficient and Resource Allocation in Cloud Computing Using QT-DNN and Binary Bird Swarm Optimization", Computers, Materials & Continua, vol. 85, no. 1, 2025.
[36] Y. Fu, D. Liu, J. Chen, and L. He, "Secretary bird optimization algorithm: a new metaheuristic for solving global optimization problems", Artificial Intelligence Review, vol. 57, no. 5, p. 123, 2023.

Authors

Dr. Mohammed A. S. Mosleh is currently serving as the Vice Dean for Academic Affairs and as Assistant Professor in the Department of Computer Science and Engineering at Al-Fayha Private College, Jubail Industrial City, Saudi Arabia. With over 10 years of teaching experience. He holds a Bachelor of Science (B.Sc.) degree in Computer Information System from Saba University, and both a Master of Science (M.Sc.) and (PhD) in Computer Science from Bharathiar University, Coimbatore, India. His research interest includes Cloud Computing, Image Processing, Task Scheduling, Resource Management and Wireless Sensor Networks. He has presented papers at national and international conferences and published research articles in peer-reviewed journals.

How to cite this article:

Mohammed A. S. Mosleh, "Carbon and Energy-Aware Cloud Resource Management and Task Scheduling Framework", International Journal of Computer Networks and Applications (IJCNA), 13(1), PP: 220-239, 2026, DOI: 10.22247/ijcna/2026/12.